Analog computations: past and future?

Dima Grigoriev (Lille)

CNRS

22/04/2014, St.Petersburg

Concept and history

As it is typical in philosophy I pose more questions than give answers.

In the absence of a formal definition under an analog computer usually is meant a physical device in which some physical values before executing a physical process are treated as an input, and some physical values after executing the process are treated as an output.

Analog computers appeared much earlier than the modern digital ones. One of the first was the astrolabe invented in Greece around II century BC. They were used mainly to calculate astronomical positions.

I don't dwell much on the history: what survived till our days is the slide rule invented in XVII century after logarithms were introduced.

In XIX century first *integrators* and *differential analyzers* appeared that could calculate the area and solve differential equations.

Concept and history

As it is typical in philosophy I pose more questions than give answers.

In the absence of a formal definition under an analog computer usually is meant a physical device in which some physical values before executing a physical process are treated as an input, and some physical values after executing the process are treated as an output.

Analog computers appeared much earlier than the modern digital ones. One of the first was the astrolabe invented in Greece around II century BC. They were used mainly to calculate astronomical positions.

I don't dwell much on the history: what survived till our days is the slide rule invented in XVII century after logarithms were introduced.

In XIX century first *integrators* and *differential analyzers* appeared that could calculate the area and solve differential equations.

Concept and history

As it is typical in philosophy I pose more questions than give answers.

In the absence of a formal definition under an analog computer usually is meant a physical device in which some physical values before executing a physical process are treated as an input, and some physical values after executing the process are treated as an output.

Analog computers appeared much earlier than the modern digital ones. One of the first was the astrolabe invented in Greece around II century BC. They were used mainly to calculate astronomical positions.

I don't dwell much on the history: what survived till our days is the slide rule invented in XVII century after logarithms were introduced.

In XIX century first *integrators* and *differential analyzers* appeared that could calculate the area and solve differential equations.

Concept and history

As it is typical in philosophy I pose more questions than give answers.

In the absence of a formal definition under an analog computer usually is meant a physical device in which some physical values before executing a physical process are treated as an input, and some physical values after executing the process are treated as an output.

Analog computers appeared much earlier than the modern digital ones. One of the first was the astrolabe invented in Greece around II century BC. They were used mainly to calculate astronomical positions.

I don't dwell much on the history: what survived till our days is the slide rule invented in XVII century after logarithms were introduced.

In XIX century first *integrators* and *differential analyzers* appeared that could calculate the area and solve differential equations.

Concept and history

As it is typical in philosophy I pose more questions than give answers.

In the absence of a formal definition under an analog computer usually is meant a physical device in which some physical values before executing a physical process are treated as an input, and some physical values after executing the process are treated as an output.

Analog computers appeared much earlier than the modern digital ones. One of the first was the astrolabe invented in Greece around II century BC. They were used mainly to calculate astronomical positions.

I don't dwell much on the history: what survived till our days is the slide rule invented in XVII century after logarithms were introduced.

In XIX century first *integrators* and *differential analyzers* appeared that could calculate the area and solve differential equations.

Concept and history

As it is typical in philosophy I pose more questions than give answers.

In the absence of a formal definition under an analog computer usually is meant a physical device in which some physical values before executing a physical process are treated as an input, and some physical values after executing the process are treated as an output.

Analog computers appeared much earlier than the modern digital ones. One of the first was the astrolabe invented in Greece around II century BC. They were used mainly to calculate astronomical positions.

I don't dwell much on the history: what survived till our days is the slide rule invented in XVII century after logarithms were introduced.

In XIX century first *integrators* and *differential analyzers* appeared that could calculate the area and solve differential equations.

Concept and history

As it is typical in philosophy I pose more questions than give answers.

In the absence of a formal definition under an analog computer usually is meant a physical device in which some physical values before executing a physical process are treated as an input, and some physical values after executing the process are treated as an output.

Analog computers appeared much earlier than the modern digital ones. One of the first was the astrolabe invented in Greece around II century BC. They were used mainly to calculate astronomical positions.

I don't dwell much on the history: what survived till our days is the slide rule invented in XVII century after logarithms were introduced.

In XIX century first *integrators* and *differential analyzers* appeared that could calculate the area and solve differential equations.

Linkages

Also in XIX century *linkages* were popular, in particular one of them due to P.L. Chebyshev. Linkage consists of several links on the plane connected by articulations. It was proved that with the help of suitable linkages one could draw an arbitrary algebraic curve.

In other words, linkages are able to emulate arithmetic, so any polynomial: say, a coordinate of a point y is a certain polynomial in a coordinate of a point x, thus conversely the coordinate of x is an algebraic function in the coordinate of y.

The latter observation was exploited by N. Mnev who has proved that classifications of linkages and arrangements of lines on the plane are apparently, unfeasible since they both lead to classifications of real semialgebraic sets (viewed as hopeless).

Linkages

Also in XIX century *linkages* were popular, in particular one of them due to P.L. Chebyshev. Linkage consists of several links on the plane connected by articulations. It was proved that with the help of suitable linkages one could draw an arbitrary algebraic curve.

In other words, linkages are able to emulate arithmetic, so any polynomial: say, a coordinate of a point y is a certain polynomial in a coordinate of a point x, thus conversely the coordinate of x is an algebraic function in the coordinate of y.

The latter observation was exploited by N. Mnev who has proved that classifications of linkages and arrangements of lines on the plane are apparently, unfeasible since they both lead to classifications of real semialgebraic sets (viewed as hopeless).

Linkages

Also in XIX century *linkages* were popular, in particular one of them due to P.L. Chebyshev. Linkage consists of several links on the plane connected by articulations. It was proved that with the help of suitable linkages one could draw an arbitrary algebraic curve.

In other words, linkages are able to emulate arithmetic, so any polynomial: say, a coordinate of a point y is a certain polynomial in a coordinate of a point x, thus conversely the coordinate of x is an algebraic function in the coordinate of y.

The latter observation was exploited by N. Mnev who has proved that classifications of linkages and arrangements of lines on the plane are apparently, unfeasible since they both lead to classifications of real semialgebraic sets (viewed as hopeless).

Linkages

Also in XIX century *linkages* were popular, in particular one of them due to P.L. Chebyshev. Linkage consists of several links on the plane connected by articulations. It was proved that with the help of suitable linkages one could draw an arbitrary algebraic curve.

In other words, linkages are able to emulate arithmetic, so any polynomial: say, a coordinate of a point y is a certain polynomial in a coordinate of a point x, thus conversely the coordinate of x is an algebraic function in the coordinate of y.

The latter observation was exploited by N. Mnev who has proved that classifications of linkages and arrangements of lines on the plane are apparently, unfeasible since they both lead to classifications of real semialgebraic sets (viewed as hopeless).

Linkages

Also in XIX century *linkages* were popular, in particular one of them due to P.L. Chebyshev. Linkage consists of several links on the plane connected by articulations. It was proved that with the help of suitable linkages one could draw an arbitrary algebraic curve.

In other words, linkages are able to emulate arithmetic, so any polynomial: say, a coordinate of a point y is a certain polynomial in a coordinate of a point x, thus conversely the coordinate of x is an algebraic function in the coordinate of y.

The latter observation was exploited by N. Mnev who has proved that classifications of linkages and arrangements of lines on the plane are apparently, unfeasible since they both lead to classifications of real semialgebraic sets (viewed as hopeless).

Linkages

Also in XIX century *linkages* were popular, in particular one of them due to P.L. Chebyshev. Linkage consists of several links on the plane connected by articulations. It was proved that with the help of suitable linkages one could draw an arbitrary algebraic curve.

In other words, linkages are able to emulate arithmetic, so any polynomial: say, a coordinate of a point y is a certain polynomial in a coordinate of a point x, thus conversely the coordinate of x is an algebraic function in the coordinate of y.

The latter observation was exploited by N. Mnev who has proved that classifications of linkages and arrangements of lines on the plane are apparently, unfeasible since they both lead to classifications of real semialgebraic sets (viewed as hopeless).

Linkages

Also in XIX century *linkages* were popular, in particular one of them due to P.L. Chebyshev. Linkage consists of several links on the plane connected by articulations. It was proved that with the help of suitable linkages one could draw an arbitrary algebraic curve.

In other words, linkages are able to emulate arithmetic, so any polynomial: say, a coordinate of a point y is a certain polynomial in a coordinate of a point x, thus conversely the coordinate of x is an algebraic function in the coordinate of y.

The latter observation was exploited by N. Mnev who has proved that classifications of linkages and arrangements of lines on the plane are apparently, unfeasible since they both lead to classifications of real semialgebraic sets (viewed as hopeless).

Integrators, neural networks

In XX century integrators were utilized till 60-s, in Russia there was in use, for example the *water integrator* based on the law of Archimedus (the latter allows one to compute the volume of an arbitrary 3-dimension body), but more efficient integrators relied on electrical circuits composed from capacitors and resistors.

More recently, neural networks in which gates use arithmetic with real numbers (electrical activity of neurons) were considered as prospective for analog computations.

Also biological computers were developed in which every cell of a biological system plays a role of a processor, this allows a parallelization, otherwise they work (very slow) as usual computers.

There is a so-called Zeno's phenomenon when an analog computation within a finite interval of time fulfils infinite number of discrete steps. Definitely, such models do not make sense, the models we consider, avoid Zeno's phenomenon. In this case the continuous time T corresponds informally to the discrete moment $\lceil T \rceil$.

Integrators, neural networks

In XX century integrators were utilized till 60-s, in Russia there was in use, for example the *water integrator* based on the law of Archimedus (the latter allows one to compute the volume of an arbitrary 3-dimension body), but more efficient integrators relied on electrical circuits composed from capacitors and resistors.

More recently, neural networks in which gates use arithmetic with real numbers (electrical activity of neurons) were considered as prospective for analog computations.

Also biological computers were developed in which every cell of a biological system plays a role of a processor, this allows a parallelization, otherwise they work (very slow) as usual computers.

There is a so-called Zeno's phenomenon when an analog computation within a finite interval of time fulfils infinite number of discrete steps. Definitely, such models do not make sense, the models we consider, avoid Zeno's phenomenon. In this case the continuous time T corresponds informally to the discrete moment $\lceil T \rceil$.

Integrators, neural networks

In XX century integrators were utilized till 60-s, in Russia there was in use, for example the *water integrator* based on the law of Archimedus (the latter allows one to compute the volume of an arbitrary 3-dimension body), but more efficient integrators relied on electrical circuits composed from capacitors and resistors.

More recently, neural networks in which gates use arithmetic with real numbers (electrical activity of neurons) were considered as prospective for analog computations.

Also biological computers were developed in which every cell of a biological system plays a role of a processor, this allows a parallelization, otherwise they work (very slow) as usual computers.

There is a so-called Zeno's phenomenon when an analog computation within a finite interval of time fulfils infinite number of discrete steps. Definitely, such models do not make sense, the models we consider, avoid Zeno's phenomenon. In this case the continuous time T corresponds informally to the discrete moment $[T]$.

Integrators, neural networks

In XX century integrators were utilized till 60-s, in Russia there was in use, for example the *water integrator* based on the law of Archimedus (the latter allows one to compute the volume of an arbitrary 3-dimension body), but more efficient integrators relied on electrical circuits composed from capacitors and resistors.

More recently, neural networks in which gates use arithmetic with real numbers (electrical activity of neurons) were considered as prospective for analog computations.

Also biological computers were developed in which every cell of a biological system plays a role of a processor, this allows a parallelization, otherwise they work (very slow) as usual computers.

There is a so-called Zeno's phenomenon when an analog computation within a finite interval of time fulfils infinite number of discrete steps. Definitely, such models do not make sense, the models we consider, avoid Zeno's phenomenon. In this case the continuous time T corresponds informally to the discrete moment $[T]$.

Integrators, neural networks

In XX century integrators were utilized till 60-s, in Russia there was in use, for example the *water integrator* based on the law of Archimedus (the latter allows one to compute the volume of an arbitrary 3-dimension body), but more efficient integrators relied on electrical circuits composed from capacitors and resistors.

More recently, neural networks in which gates use arithmetic with real numbers (electrical activity of neurons) were considered as prospective for analog computations.

Also biological computers were developed in which every cell of a biological system plays a role of a processor, this allows a parallelization, otherwise they work (very slow) as usual computers.

There is a so-called Zeno's phenomenon when an analog computation within a finite interval of time fulfils infinite number of discrete steps. Definitely, such models do not make sense, the models we consider, avoid Zeno's phenomenon. In this case the continuous time T corresponds informally to the discrete moment $\lceil T \rceil$.

Integrators, neural networks

In XX century integrators were utilized till 60-s, in Russia there was in use, for example the *water integrator* based on the law of Archimedus (the latter allows one to compute the volume of an arbitrary 3-dimension body), but more efficient integrators relied on electrical circuits composed from capacitors and resistors.

More recently, neural networks in which gates use arithmetic with real numbers (electrical activity of neurons) were considered as prospective for analog computations.

Also biological computers were developed in which every cell of a biological system plays a role of a processor, this allows a parallelization, otherwise they work (very slow) as usual computers.

There is a so-called Zeno's phenomenon when an analog computation within a finite interval of time fulfils infinite number of discrete steps. Definitely, such models do not make sense, the models we consider, avoid Zeno's phenomenon. In this case the continuous time T corresponds informally to the discrete moment $\lceil T \rceil$.

Integrators, neural networks

In XX century integrators were utilized till 60-s, in Russia there was in use, for example the *water integrator* based on the law of Archimedus (the latter allows one to compute the volume of an arbitrary 3-dimension body), but more efficient integrators relied on electrical circuits composed from capacitors and resistors.

More recently, neural networks in which gates use arithmetic with real numbers (electrical activity of neurons) were considered as prospective for analog computations.

Also biological computers were developed in which every cell of a biological system plays a role of a processor, this allows a parallelization, otherwise they work (very slow) as usual computers.

There is a so-called Zeno's phenomenon when an analog computation within a finite interval of time fulfils infinite number of discrete steps. Definitely, such models do not make sense, the models we consider, avoid Zeno's phenomenon. In this case the continuous time T corresponds informally to the discrete moment $\lfloor T \rfloor$.

Integrators, neural networks

In XX century integrators were utilized till 60-s, in Russia there was in use, for example the *water integrator* based on the law of Archimedus (the latter allows one to compute the volume of an arbitrary 3-dimension body), but more efficient integrators relied on electrical circuits composed from capacitors and resistors.

More recently, neural networks in which gates use arithmetic with real numbers (electrical activity of neurons) were considered as prospective for analog computations.

Also biological computers were developed in which every cell of a biological system plays a role of a processor, this allows a parallelization, otherwise they work (very slow) as usual computers.

There is a so-called Zeno's phenomenon when an analog computation within a finite interval of time fulfils infinite number of discrete steps.

Definitely, such models do not make sense, the models we consider, avoid Zeno's phenomenon. In this case the continuous time T corresponds informally to the discrete moment $\lceil T \rceil$.

Integrators, neural networks

In XX century integrators were utilized till 60-s, in Russia there was in use, for example the *water integrator* based on the law of Archimedus (the latter allows one to compute the volume of an arbitrary 3-dimension body), but more efficient integrators relied on electrical circuits composed from capacitors and resistors.

More recently, neural networks in which gates use arithmetic with real numbers (electrical activity of neurons) were considered as prospective for analog computations.

Also biological computers were developed in which every cell of a biological system plays a role of a processor, this allows a parallelization, otherwise they work (very slow) as usual computers.

There is a so-called Zeno's phenomenon when an analog computation within a finite interval of time fulfils infinite number of discrete steps. Definitely, such models do not make sense, the models we consider, avoid Zeno's phenomenon. In this case the continuous time T corresponds informally to the discrete moment $\lceil T \rceil$.

Integrators, neural networks

In XX century integrators were utilized till 60-s, in Russia there was in use, for example the *water integrator* based on the law of Archimedus (the latter allows one to compute the volume of an arbitrary 3-dimension body), but more efficient integrators relied on electrical circuits composed from capacitors and resistors.

More recently, neural networks in which gates use arithmetic with real numbers (electrical activity of neurons) were considered as prospective for analog computations.

Also biological computers were developed in which every cell of a biological system plays a role of a processor, this allows a parallelization, otherwise they work (very slow) as usual computers.

There is a so-called Zeno's phenomenon when an analog computation within a finite interval of time fulfils infinite number of discrete steps. Definitely, such models do not make sense, the models we consider, avoid Zeno's phenomenon. In this case the continuous time T corresponds informally to the discrete moment $[T]$.

Church's thesis for analog computers?

In any case an analog computer is a device which relies on a certain physical or chemical law, so if x is a real input then $y = f(x)$ is a real output where f is a function describing the underlying law.

Instead of calculations by a digital computer, an analog computer requires a measurement of y, and therefore the quality of the output depends on precision of the measurement. Also it is more difficult to control errors emerging due to possible noise in an analog computers rather than in digital ones. Say, if a physical process on which an analog computer is based is unstable it could lead to a big error of the output as a result of a small error in the input.

One can convert f into an integer function $[f] : \mathbf{Z} \to \mathbf{Z}$ where $[f](x) := [f(x)]$. Then a question arises which can be called

Church's thesis for analog computers: *Is* $[f]$ *a recursive function?*

On the other hand, O. Burnez, D. Graca, A. Pouly have designed analog computers based on differential dynamical systems which can compute any recursive function.

Church's thesis for analog computers?

In any case an analog computer is a device which relies on a certain physical or chemical law, so if x is a real input then $y = f(x)$ is a real output where f is a function describing the underlying law.

Instead of calculations by a digital computer, an analog computer requires a measurement of y, and therefore the quality of the output depends on precision of the measurement. Also it is more difficult to control errors emerging due to possible noise in an analog computers rather than in digital ones. Say, if a physical process on which an analog computer is based is unstable it could lead to a big error of the output as a result of a small error in the input.

One can convert f into an integer function $[f] : \mathbf{Z} \to \mathbf{Z}$ where $[f](x) := [f(x)]$. Then a question arises which can be called

Church's thesis for analog computers: *Is $[f]$ a recursive function?*

On the other hand, O. Burnez, D. Graca, A. Pouly have designed analog computers based on differential dynamical systems which can compute any recursive function.

Church's thesis for analog computers?

In any case an analog computer is a device which relies on a certain physical or chemical law, so if x is a real input then $y = f(x)$ is a real output where f is a function describing the underlying law.

Instead of calculations by a digital computer, an analog computer requires a measurement of y, and therefore the quality of the output depends on precision of the measurement. Also it is more difficult to control errors emerging due to possible noise in an analog computers rather than in digital ones. Say, if a physical process on which an analog computer is based is unstable it could lead to a big error of the output as a result of a small error in the input.

One can convert f into an integer function $[f] : \mathbf{Z} \to \mathbf{Z}$ where $[f](x) := [f(x)]$. Then a question arises which can be called

Church's thesis for analog computers: *Is* $[f]$ *a recursive function?*

On the other hand, O. Burnez, D. Graca, A. Pouly have designed analog computers based on differential dynamical systems which can compute any recursive function.

Church's thesis for analog computers?

In any case an analog computer is a device which relies on a certain physical or chemical law, so if x is a real input then $y = f(x)$ is a real output where f is a function describing the underlying law.

Instead of calculations by a digital computer, an analog computer requires a measurement of y, and therefore the quality of the output depends on precision of the measurement. Also it is more difficult to control errors emerging due to possible noise in an analog computers rather than in digital ones. Say, if a physical process on which an analog computer is based is unstable it could lead to a big error of the output as a result of a small error in the input.

One can convert f into an integer function $[f] : \mathbf{Z} \to \mathbf{Z}$ where $[f](x) := [f(x)]$. Then a question arises which can be called

Church's thesis for analog computers: *Is* $[f]$ *a recursive function?*

On the other hand, O. Burnez, D. Graca, A. Pouly have designed analog computers based on differential dynamical systems which can compute any recursive function.

Church's thesis for analog computers?

In any case an analog computer is a device which relies on a certain physical or chemical law, so if x is a real input then $y = f(x)$ is a real output where f is a function describing the underlying law.

Instead of calculations by a digital computer, an analog computer requires a measurement of y, and therefore the quality of the output depends on precision of the measurement. Also it is more difficult to control errors emerging due to possible noise in an analog computers rather than in digital ones. Say, if a physical process on which an analog computer is based is unstable it could lead to a big error of the output as a result of a small error in the input.

One can convert f into an integer function $[f] : \mathbf{Z} \to \mathbf{Z}$ where $[f](x) := [f(x)]$. Then a question arises which can be called

Church's thesis for analog computers: *Is* $[f]$ *a recursive function?*

On the other hand, O. Burnez, D. Graca, A. Pouly have designed analog computers based on differential dynamical systems which can compute any recursive function.

Church's thesis for analog computers?

In any case an analog computer is a device which relies on a certain physical or chemical law, so if x is a real input then $y = f(x)$ is a real output where f is a function describing the underlying law.

Instead of calculations by a digital computer, an analog computer requires a measurement of y, and therefore the quality of the output depends on precision of the measurement. Also it is more difficult to control errors emerging due to possible noise in an analog computers rather than in digital ones. Say, if a physical process on which an analog computer is based is unstable it could lead to a big error of the output as a result of a small error in the input.

One can convert f into an integer function $[f] : \mathbf{Z} \to \mathbf{Z}$ where $[f](x) := [f(x)]$. Then a question arises which can be called

Church's thesis for analog computers: *Is $[f]$ a recursive function?*

On the other hand, O. Burnez, D. Graca, A. Pouly have designed analog computers based on differential dynamical systems which can compute any recursive function.

Church's thesis for analog computers?

In any case an analog computer is a device which relies on a certain physical or chemical law, so if x is a real input then $y = f(x)$ is a real output where f is a function describing the underlying law.

Instead of calculations by a digital computer, an analog computer requires a measurement of y, and therefore the quality of the output depends on precision of the measurement. Also it is more difficult to control errors emerging due to possible noise in an analog computers rather than in digital ones. Say, if a physical process on which an analog computer is based is unstable it could lead to a big error of the output as a result of a small error in the input.

One can convert f into an integer function $[f] : \mathbf{Z} \rightarrow \mathbf{Z}$ where $[f](x) := [f(x)]$. Then a question arises which can be called

Church's thesis for analog computers: *Is $[f]$ a recursive function?*

On the other hand, O. Burnez, D. Graca, A. Pouly have designed analog computers based on differential dynamical systems which can compute any recursive function.

Church's thesis for analog computers?

In any case an analog computer is a device which relies on a certain physical or chemical law, so if x is a real input then $y = f(x)$ is a real output where f is a function describing the underlying law.

Instead of calculations by a digital computer, an analog computer requires a measurement of y, and therefore the quality of the output depends on precision of the measurement. Also it is more difficult to control errors emerging due to possible noise in an analog computers rather than in digital ones. Say, if a physical process on which an analog computer is based is unstable it could lead to a big error of the output as a result of a small error in the input.

One can convert f into an integer function $[f] : \mathbf{Z} \to \mathbf{Z}$ where $[f](x) := [f(x)]$. Then a question arises which can be called

Church's thesis for analog computers: *Is* $[f]$ *a recursive function?*

On the other hand, O. Burnez, D. Graca, A. Pouly have designed analog computers based on differential dynamical systems which can compute any recursive function.

Church's thesis for analog computers?

In any case an analog computer is a device which relies on a certain physical or chemical law, so if x is a real input then $y = f(x)$ is a real output where f is a function describing the underlying law.

Instead of calculations by a digital computer, an analog computer requires a measurement of y, and therefore the quality of the output depends on precision of the measurement. Also it is more difficult to control errors emerging due to possible noise in an analog computers rather than in digital ones. Say, if a physical process on which an analog computer is based is unstable it could lead to a big error of the output as a result of a small error in the input.

One can convert f into an integer function $[f] : \mathbf{Z} \to \mathbf{Z}$ where $[f](x) := [f(x)]$. Then a question arises which can be called

Church's thesis for analog computers: *Is* $[f]$ *a recursive function?*

On the other hand, O. Burnez, D. Graca, A. Pouly have designed analog computers based on differential dynamical systems which can compute any recursive function.

Universal analog computer?

To prove Church's thesis for analog computers one has to define the latter formally (so far, just some particular constructions of analog computers were suggested).

On the other hand, to refute the Church's thesis it would be sufficient to produce a reasonable analog computer which computes a non-recursive function. But even if the latter was possible, still a question would arise on a universal analog computer (note that the classical Church's thesis besides the vague conjecture that any algorithm can be realized by a Turing machine, comprises the construction of a universal Turing machine):

Is there a universal analog computer?

Universal analog computer?

To prove Church's thesis for analog computers one has to define the latter formally (so far, just some particular constructions of analog computers were suggested).

On the other hand, to refute the Church's thesis it would be sufficient to produce a reasonable analog computer which computes a non-recursive function. But even if the latter was possible, still a question would arise on a universal analog computer (note that the classical Church's thesis besides the vague conjecture that any algorithm can be realized by a Turing machine, comprises the construction of a universal Turing machine):

Is there a universal analog computer?

Universal analog computer?

To prove Church's thesis for analog computers one has to define the latter formally (so far, just some particular constructions of analog computers were suggested).

On the other hand, to refute the Church's thesis it would be sufficient to produce a reasonable analog computer which computes a non-recursive function. But even if the latter was possible, still a question would arise on a universal analog computer (note that the classical Church's thesis besides the vague conjecture that any algorithm can be realized by a Turing machine, comprises the construction of a universal Turing machine):

Is there a universal analog computer?

Universal analog computer?

To prove Church's thesis for analog computers one has to define the latter formally (so far, just some particular constructions of analog computers were suggested).

On the other hand, to refute the Church's thesis it would be sufficient to produce a reasonable analog computer which computes a non-recursive function. But even if the latter was possible, still a question would arise on a universal analog computer (note that the classical Church's thesis besides the vague conjecture that any algorithm can be realized by a Turing machine, comprises the construction of a universal Turing machine):

Is there a universal analog computer?

Universal analog computer?

To prove Church's thesis for analog computers one has to define the latter formally (so far, just some particular constructions of analog computers were suggested).

On the other hand, to refute the Church's thesis it would be sufficient to produce a reasonable analog computer which computes a non-recursive function. But even if the latter was possible, still a question would arise on a universal analog computer (note that the classical Church's thesis besides the vague conjecture that any algorithm can be realized by a Turing machine, comprises the construction of a universal Turing machine):

Is there a universal analog computer?

Universal analog computer?

To prove Church's thesis for analog computers one has to define the latter formally (so far, just some particular constructions of analog computers were suggested).

On the other hand, to refute the Church's thesis it would be sufficient to produce a reasonable analog computer which computes a non-recursive function. But even if the latter was possible, still a question would arise on a universal analog computer (note that the classical Church's thesis besides the vague conjecture that any algorithm can be realized by a Turing machine, comprises the construction of a universal Turing machine):

Is there a universal analog computer?

Universal analog computer?

To prove Church's thesis for analog computers one has to define the latter formally (so far, just some particular constructions of analog computers were suggested).

On the other hand, to refute the Church's thesis it would be sufficient to produce a reasonable analog computer which computes a non-recursive function. But even if the latter was possible, still a question would arise on a universal analog computer (note that the classical Church's thesis besides the vague conjecture that any algorithm can be realized by a Turing machine, comprises the construction of a universal Turing machine):

Is there a universal analog computer?

Turing machines = General Purpose Analog Computers

C. Shannon introduced General Purpose Analog Computers (GPAC). They compute real functions being solutions of autonomous dynamical systems with polynomial coefficients. In GPAC the euclidean norm of solutions plays the role of the space of computations. In particular, the space could be exponentially larger than the time (unlike Turing machines for which the space always does not exceed the time).

O. Burnez, D. Graca, A. Pouly have established the equivalence up to a polynomial in pairs of resources (*time, space*) between Turing machines and GPAC. In particular, the set of functions computed by GPAC coincides with the set of recursive functions. For our further discussion on P-NP problem we can conclude that GPAC can't solve NP-hard problems in polynomial time (unless $P \neq NP$).

Turing machines = General Purpose Analog Computers

C. Shannon introduced General Purpose Analog Computers (GPAC). They compute real functions being solutions of autonomous dynamical systems with polynomial coefficients. In GPAC the euclidean norm of solutions plays the role of the space of computations. In particular, the space could be exponentially larger than the time (unlike Turing machines for which the space always does not exceed the time).

O. Burnez, D. Graca, A. Pouly have established the equivalence up to a polynomial in pairs of resources (*time, space*) between Turing machines and GPAC. In particular, the set of functions computed by GPAC coincides with the set of recursive functions. For our further discussion on P-NP problem we can conclude that GPAC can't solve NP-hard problems in polynomial time (unless $P \neq NP$).

Turing machines = General Purpose Analog Computers

C. Shannon introduced General Purpose Analog Computers (GPAC). They compute real functions being solutions of autonomous dynamical systems with polynomial coefficients. In GPAC the euclidean norm of solutions plays the role of the space of computations. In particular, the space could be exponentially larger than the time (unlike Turing machines for which the space always does not exceed the time).

O. Burnez, D. Graca, A. Pouly have established the equivalence up to a polynomial in pairs of resources (*time, space*) between Turing machines and GPAC. In particular, the set of functions computed by GPAC coincides with the set of recursive functions. For our further discussion on P-NP problem we can conclude that GPAC can't solve NP-hard problems in polynomial time (unless $P \neq NP$).

Turing machines = General Purpose Analog Computers

C. Shannon introduced General Purpose Analog Computers (GPAC). They compute real functions being solutions of autonomous dynamical systems with polynomial coefficients. In GPAC the euclidean norm of solutions plays the role of the space of computations. In particular, the space could be exponentially larger than the time (unlike Turing machines for which the space always does not exceed the time).

O. Burnez, D. Graca, A. Pouly have established the equivalence up to a polynomial in pairs of resources (*time, space*) between Turing machines and GPAC. In particular, the set of functions computed by GPAC coincides with the set of recursive functions. For our further discussion on P-NP problem we can conclude that GPAC can't solve NP-hard problems in polynomial time (unless $P \neq NP$).

Turing machines = General Purpose Analog Computers

C. Shannon introduced General Purpose Analog Computers (GPAC). They compute real functions being solutions of autonomous dynamical systems with polynomial coefficients. In GPAC the euclidean norm of solutions plays the role of the space of computations. In particular, the space could be exponentially larger than the time (unlike Turing machines for which the space always does not exceed the time).

O. Burnez, D. Graca, A. Pouly have established the equivalence up to a polynomial in pairs of resources (*time, space*) between Turing machines and GPAC. In particular, the set of functions computed by GPAC coincides with the set of recursive functions. For our further discussion on P-NP problem we can conclude that GPAC can't solve NP-hard problems in polynomial time (unless $P \neq NP$).

Turing machines = General Purpose Analog Computers

C. Shannon introduced General Purpose Analog Computers (GPAC). They compute real functions being solutions of autonomous dynamical systems with polynomial coefficients. In GPAC the euclidean norm of solutions plays the role of the space of computations. In particular, the space could be exponentially larger than the time (unlike Turing machines for which the space always does not exceed the time).

O. Burnez, D. Graca, A. Pouly have established the equivalence up to a polynomial in pairs of resources (*time, space*) between Turing machines and GPAC. In particular, the set of functions computed by GPAC coincides with the set of recursive functions. For our further discussion on P-NP problem we can conclude that GPAC can't solve NP-hard problems in polynomial time (unless $P \neq NP$).

Turing machines = General Purpose Analog Computers

C. Shannon introduced General Purpose Analog Computers (GPAC). They compute real functions being solutions of autonomous dynamical systems with polynomial coefficients. In GPAC the euclidean norm of solutions plays the role of the space of computations. In particular, the space could be exponentially larger than the time (unlike Turing machines for which the space always does not exceed the time).

O. Burnez, D. Graca, A. Pouly have established the equivalence up to a polynomial in pairs of resources (*time, space*) between Turing machines and GPAC. In particular, the set of functions computed by GPAC coincides with the set of recursive functions. For our further discussion on P-NP problem we can conclude that GPAC can't solve NP-hard problems in polynomial time (unless $P \neq NP$).

Turing machines = General Purpose Analog Computers

C. Shannon introduced General Purpose Analog Computers (GPAC). They compute real functions being solutions of autonomous dynamical systems with polynomial coefficients. In GPAC the euclidean norm of solutions plays the role of the space of computations. In particular, the space could be exponentially larger than the time (unlike Turing machines for which the space always does not exceed the time).

O. Burnez, D. Graca, A. Pouly have established the equivalence up to a polynomial in pairs of resources (*time, space*) between Turing machines and GPAC. In particular, the set of functions computed by GPAC coincides with the set of recursive functions. For our further discussion on P-NP problem we can conclude that GPAC can't solve NP-hard problems in polynomial time (unless $P \neq NP$).

Blum-Shub-Smale machines

An attempt of a mathematical formalism of analog computers provide Blum-Shub-Smale (BSS) machines. They are able to perform arithmetic operations with reals and make a branching according to the sign of a result. Sometimes arbitrary real constants are admitted in BSS machines and also taking the integer part. Then BSS machine can be treated as computing a function from integers to integers, and M. Shub, S. Smale have noticed that such machines can compute non-recursive functions (due to arbitrariness of invoked real constants).

In particular, a question arises in connection with Church's thesis for analog computers: can uncomputable reals occur in physical laws?

Uncomputable solutions of differential equations

M. Pour-El, J. Richards have produced a system of differential equations with uncomputable solutions. If one could design an analog computation with the behaviour described by such a system of differential equations then one would refute Church's thesis for analog computers.

Blum-Shub-Smale machines

An attempt of a mathematical formalism of analog computers provide Blum-Shub-Smale (BSS) machines. They are able to perform arithmetic operations with reals and make a branching according to the sign of a result. Sometimes arbitrary real constants are admitted in BSS machines and also taking the integer part. Then BSS machine can be treated as computing a function from integers to integers, and M. Shub, S. Smale have noticed that such machines can compute non-recursive functions (due to arbitrariness of invoked real constants).

In particular, a question arises in connection with Church's thesis for analog computers: can uncomputable reals occur in physical laws?

Uncomputable solutions of differential equations

M. Pour-El, J. Richards have produced a system of differential equations with uncomputable solutions. If one could design an analog computation with the behaviour described by such a system of differential equations then one would refute Church's thesis for analog computers.

Blum-Shub-Smale machines

An attempt of a mathematical formalism of analog computers provide Blum-Shub-Smale (BSS) machines. They are able to perform arithmetic operations with reals and make a branching according to the sign of a result. Sometimes arbitrary real constants are admitted in BSS machines and also taking the integer part. Then BSS machine can be treated as computing a function from integers to integers, and M. Shub, S. Smale have noticed that such machines can compute non-recursive functions (due to arbitrariness of invoked real constants).

In particular, a question arises in connection with Church's thesis for analog computers: can uncomputable reals occur in physical laws?

Uncomputable solutions of differential equations

M. Pour-El, J. Richards have produced a system of differential equations with uncomputable solutions. If one could design an analog computation with the behaviour described by such a system of differential equations then one would refute Church's thesis for analog computers.

Blum-Shub-Smale machines

An attempt of a mathematical formalism of analog computers provide Blum-Shub-Smale (BSS) machines. They are able to perform arithmetic operations with reals and make a branching according to the sign of a result. Sometimes arbitrary real constants are admitted in BSS machines and also taking the integer part. Then BSS machine can be treated as computing a function from integers to integers, and M. Shub, S. Smale have noticed that such machines can compute non-recursive functions (due to arbitrariness of invoked real constants).

In particular, a question arises in connection with Church's thesis for analog computers: can uncomputable reals occur in physical laws?

Uncomputable solutions of differential equations

M. Pour-El, J. Richards have produced a system of differential equations with uncomputable solutions. If one could design an analog computation with the behaviour described by such a system of differential equations then one would refute Church's thesis for analog computers.

Blum-Shub-Smale machines

An attempt of a mathematical formalism of analog computers provide Blum-Shub-Smale (BSS) machines. They are able to perform arithmetic operations with reals and make a branching according to the sign of a result. Sometimes arbitrary real constants are admitted in BSS machines and also taking the integer part. Then BSS machine can be treated as computing a function from integers to integers, and M. Shub, S. Smale have noticed that such machines can compute non-recursive functions (due to arbitrariness of invoked real constants).

In particular, a question arises in connection with Church's thesis for analog computers: can uncomputable reals occur in physical laws?

Uncomputable solutions of differential equations

M. Pour-El, J. Richards have produced a system of differential equations with uncomputable solutions. If one could design an analog computation with the behaviour described by such a system of differential equations then one would refute Church's thesis for analog computers.

Blum-Shub-Smale machines

An attempt of a mathematical formalism of analog computers provide Blum-Shub-Smale (BSS) machines. They are able to perform arithmetic operations with reals and make a branching according to the sign of a result. Sometimes arbitrary real constants are admitted in BSS machines and also taking the integer part. Then BSS machine can be treated as computing a function from integers to integers, and M. Shub, S. Smale have noticed that such machines can compute non-recursive functions (due to arbitrariness of invoked real constants).

In particular, a question arises in connection with Church's thesis for analog computers: can uncomputable reals occur in physical laws?

Uncomputable solutions of differential equations

M. Pour-El, J. Richards have produced a system of differential equations with uncomputable solutions. If one could design an analog computation with the behaviour described by such a system of differential equations then one would refute Church's thesis for analog computers.

Blum-Shub-Smale machines

An attempt of a mathematical formalism of analog computers provide Blum-Shub-Smale (BSS) machines. They are able to perform arithmetic operations with reals and make a branching according to the sign of a result. Sometimes arbitrary real constants are admitted in BSS machines and also taking the integer part. Then BSS machine can be treated as computing a function from integers to integers, and M. Shub, S. Smale have noticed that such machines can compute non-recursive functions (due to arbitrariness of invoked real constants).

In particular, a question arises in connection with Church's thesis for analog computers: can uncomputable reals occur in physical laws?

Uncomputable solutions of differential equations

M. Pour-El, J. Richards have produced a system of differential equations with uncomputable solutions. If one could design an analog computation with the behaviour described by such a system of differential equations then one would refute Church's thesis for analog computers.

Blum-Shub-Smale machines

An attempt of a mathematical formalism of analog computers provide Blum-Shub-Smale (BSS) machines. They are able to perform arithmetic operations with reals and make a branching according to the sign of a result. Sometimes arbitrary real constants are admitted in BSS machines and also taking the integer part. Then BSS machine can be treated as computing a function from integers to integers, and M. Shub, S. Smale have noticed that such machines can compute non-recursive functions (due to arbitrariness of invoked real constants).

In particular, a question arises in connection with Church's thesis for analog computers: can uncomputable reals occur in physical laws?

Uncomputable solutions of differential equations

M. Pour-El, J. Richards have produced a system of differential equations with uncomputable solutions. If one could design an analog computation with the behaviour described by such a system of differential equations then one would refute Church's thesis for analog computers.

Can NP-hard problems be solved by analog computer in polynomial time?

We say that a computational problem $A(n)$ where n is an integer, is solved by an analog computer C within time $t(\log n)$ (taking into account that $[\log n] + 1$ is the bit-size of n) if $t(\log n)$ majorates the overall time of both designing $C(n)$ and its running time. Typically (but not necessary), the running time of an analog computer is much less the time of its designing. This is the case, say, for analog computers based on electrical circuits.

Yu. Matiyasevich was one of the first who tried to answer the question

Is there an analog computer which solves an NP-hard problem within $t(\log n) < poly(\log n)$ time?

But it seems that rigorously speaking this question still remains open. There are common difficulties in diverse attempts towards answering this question.

Can NP-hard problems be solved by analog computer in polynomial time?

We say that a computational problem $A(n)$ where n is an integer, is solved by an analog computer C within time $t(\log n)$ (taking into account that $[\log n] + 1$ is the bit-size of n) if $t(\log n)$ majorates the overall time of both designing $C(n)$ and its running time. Typically (but not necessary), the running time of an analog computer is much less the time of its designing. This is the case, say, for analog computers based on electrical circuits.

Yu. Matiyasevich was one of the first who tried to answer the question

Is there an analog computer which solves an NP-hard problem within $t(\log n) < poly(\log n)$ **time?**

But it seems that rigorously speaking this question still remains open. There are common difficulties in diverse attempts towards answering this question.

Can NP-hard problems be solved by analog computer in polynomial time?

We say that a computational problem $A(n)$ where n is an integer, is solved by an analog computer C within time $t(\log n)$ (taking into account that $[\log n] + 1$ is the bit-size of n) if $t(\log n)$ majorates the overall time of both designing $C(n)$ and its running time. Typically (but not necessary), the running time of an analog computer is much less the time of its designing. This is the case, say, for analog computers based on electrical circuits.

Yu. Matiyasevich was one of the first who tried to answer the question

Is there an analog computer which solves an NP-hard problem within $t(\log n) < poly(\log n)$ time?

But it seems that rigorously speaking this question still remains open. There are common difficulties in diverse attempts towards answering this question.

Can NP-hard problems be solved by analog computer in polynomial time?

We say that a computational problem $A(n)$ where n is an integer, is solved by an analog computer C within time $t(\log n)$ (taking into account that $[\log n] + 1$ is the bit-size of n) if $t(\log n)$ majorates the overall time of both designing $C(n)$ and its running time. Typically (but not necessary), the running time of an analog computer is much less the time of its designing. This is the case, say, for analog computers based on electrical circuits.

Yu. Matiyasevich was one of the first who tried to answer the question

Is there an analog computer which solves an NP-hard problem within $t(\log n) < poly(\log n)$ time?

But it seems that rigorously speaking this question still remains open. There are common difficulties in diverse attempts towards answering this question.

Can NP-hard problems be solved by analog computer in polynomial time?

We say that a computational problem $A(n)$ where n is an integer, is solved by an analog computer C within time $t(\log n)$ (taking into account that $[\log n] + 1$ is the bit-size of n) if $t(\log n)$ majorates the overall time of both designing $C(n)$ and its running time. Typically (but not necessary), the running time of an analog computer is much less the time of its designing. This is the case, say, for analog computers based on electrical circuits.

Yu. Matiyasevich was one of the first who tried to answer the question

Is there an analog computer which solves an NP-hard problem within $t(\log n) < poly(\log n)$ time?

But it seems that rigorously speaking this question still remains open. There are common difficulties in diverse attempts towards answering this question.

Can NP-hard problems be solved by analog computer in polynomial time?

We say that a computational problem $A(n)$ where n is an integer, is solved by an analog computer C within time $t(\log n)$ (taking into account that $[\log n] + 1$ is the bit-size of n) if $t(\log n)$ majorates the overall time of both designing $C(n)$ and its running time. Typically (but not necessary), the running time of an analog computer is much less the time of its designing. This is the case, say, for analog computers based on electrical circuits.

Yu. Matiyasevich was one of the first who tried to answer the question

Is there an analog computer which solves an NP-hard problem within $t(\log n) < poly(\log n)$ time?

But it seems that rigorously speaking this question still remains open. There are common difficulties in diverse attempts towards answering this question.

Can NP-hard problems be solved by analog computer in polynomial time?

We say that a computational problem $A(n)$ where n is an integer, is solved by an analog computer C within time $t(\log n)$ (taking into account that $[\log n] + 1$ is the bit-size of n) if $t(\log n)$ majorates the overall time of both designing $C(n)$ and its running time. Typically (but not necessary), the running time of an analog computer is much less the time of its designing. This is the case, say, for analog computers based on electrical circuits.

Yu. Matiyasevich was one of the first who tried to answer the question

Is there an analog computer which solves an NP-hard problem within $t(\log n) < poly(\log n)$ **time?**

But it seems that rigorously speaking this question still remains open. There are common difficulties in diverse attempts towards answering this question.

Can NP-hard problems be solved by analog computer in polynomial time?

We say that a computational problem $A(n)$ where n is an integer, is solved by an analog computer C within time $t(\log n)$ (taking into account that $[\log n] + 1$ is the bit-size of n) if $t(\log n)$ majorates the overall time of both designing $C(n)$ and its running time. Typically (but not necessary), the running time of an analog computer is much less the time of its designing. This is the case, say, for analog computers based on electrical circuits.

Yu. Matiyasevich was one of the first who tried to answer the question

Is there an analog computer which solves an NP-hard problem within $t(\log n) < poly(\log n)$ time?

But it seems that rigorously speaking this question still remains open. There are common difficulties in diverse attempts towards answering this question.

Can NP-hard problems be solved by analog computer in polynomial time?

We say that a computational problem $A(n)$ where n is an integer, is solved by an analog computer C within time $t(\log n)$ (taking into account that $[\log n] + 1$ is the bit-size of n) if $t(\log n)$ majorates the overall time of both designing $C(n)$ and its running time. Typically (but not necessary), the running time of an analog computer is much less the time of its designing. This is the case, say, for analog computers based on electrical circuits.

Yu. Matiyasevich was one of the first who tried to answer the question

Is there an analog computer which solves an NP-hard problem within $t(\log n) < poly(\log n)$ time?

But it seems that rigorously speaking this question still remains open. There are common difficulties in diverse attempts towards answering this question.

Can NP-hard problems be solved by analog computer in polynomial time?

We say that a computational problem $A(n)$ where n is an integer, is solved by an analog computer C within time $t(\log n)$ (taking into account that $[\log n] + 1$ is the bit-size of n) if $t(\log n)$ majorates the overall time of both designing $C(n)$ and its running time. Typically (but not necessary), the running time of an analog computer is much less the time of its designing. This is the case, say, for analog computers based on electrical circuits.

Yu. Matiyasevich was one of the first who tried to answer the question

Is there an analog computer which solves an NP-hard problem within $t(\log n) < poly(\log n)$ **time?**

But it seems that rigorously speaking this question still remains open. There are common difficulties in diverse attempts towards answering this question.

Some difficulties in solving NP-hard problems by means of analog computers

The first difficulty arises when one wants to encode an integer n: it is reasonable to encode it by a value of some physical variable of a magnitude close to n, but then its "physical size" n is exponential in bit-size $\lceil \log n \rceil + 1$. That is why in analog devices integers are often encoded by their bits, while each bit is encoded in an analog way.

Another difficulty is that many analog computers bring an underlying dynamical system to a stable state, say a local minimum of a certain target function. Examples of this are the minimal-tension surface of a soap film, or the steepest gradient of a falling stone from the hill, or a minimal energy state in the Ising spin-glass model etc. But to solve an NP-hard problem one has typically to find a global minimum, and there can be an exponential number of local minima.

Some difficulties in solving NP-hard problems by means of analog computers

The first difficulty arises when one wants to encode an integer n: it is reasonable to encode it by a value of some physical variable of a magnitude close to n, but then its "physical size" n is exponential in bit-size $[\log n] + 1$. That is why in analog devices integers are often encoded by their bits, while each bit is encoded in an analog way.

Another difficulty is that many analog computers bring an underlying dynamical system to a stable state, say a local minimum of a certain target function. Examples of this are the minimal-tension surface of a soap film, or the steepest gradient of a falling stone from the hill, or a minimal energy state in the Ising spin-glass model etc. But to solve an NP-hard problem one has typically to find a global minimum, and there can be an exponential number of local minima.

Some difficulties in solving NP-hard problems by means of analog computers

The first difficulty arises when one wants to encode an integer n: it is reasonable to encode it by a value of some physical variable of a magnitude close to n, but then its ”physical size” n is exponential in bit-size $[\log n] + 1$. That is why in analog devices integers are often encoded by their bits, while each bit is encoded in an analog way.

Another difficulty is that many analog computers bring an underlying dynamical system to a stable state, say a local minimum of a certain target function. Examples of this are the minimal-tension surface of a soap film, or the steepest gradient of a falling stone from the hill, or a minimal energy state in the Ising spin-glass model etc. But to solve an NP-hard problem one has typically to find a global minimum, and there can be an exponential number of local minima.

Some difficulties in solving NP-hard problems by means of analog computers

The first difficulty arises when one wants to encode an integer n: it is reasonable to encode it by a value of some physical variable of a magnitude close to n, but then its ”physical size” n is exponential in bit-size $[\log n] + 1$. That is why in analog devices integers are often encoded by their bits, while each bit is encoded in an analog way.

Another difficulty is that many analog computers bring an underlying dynamical system to a stable state, say a local minimum of a certain target function. Examples of this are the minimal-tension surface of a soap film, or the steepest gradient of a falling stone from the hill, or a minimal energy state in the Ising spin-glass model etc. But to solve an NP-hard problem one has typically to find a global minimum, and there can be an exponential number of local minima.

Some difficulties in solving NP-hard problems by means of analog computers

The first difficulty arises when one wants to encode an integer n: it is reasonable to encode it by a value of some physical variable of a magnitude close to n, but then its ”physical size” n is exponential in bit-size $[\log n] + 1$. That is why in analog devices integers are often encoded by their bits, while each bit is encoded in an analog way.

Another difficulty is that many analog computers bring an underlying dynamical system to a stable state, say a local minimum of a certain target function. Examples of this are the minimal-tension surface of a soap film, or the steepest gradient of a falling stone from the hill, or a minimal energy state in the Ising spin-glass model etc. But to solve an NP-hard problem one has typically to find a global minimum, and there can be an exponential number of local minima.

Some difficulties in solving NP-hard problems by means of analog computers

The first difficulty arises when one wants to encode an integer n: it is reasonable to encode it by a value of some physical variable of a magnitude close to n, but then its "physical size" n is exponential in bit-size $[\log n] + 1$. That is why in analog devices integers are often encoded by their bits, while each bit is encoded in an analog way.

Another difficulty is that many analog computers bring an underlying dynamical system to a stable state, say a local minimum of a certain target function. Examples of this are the minimal-tension surface of a soap film, or the steepest gradient of a falling stone from the hill, or a minimal energy state in the Ising spin-glass model etc. But to solve an NP-hard problem one has typically to find a global minimum, and there can be an exponential number of local minima.

Some difficulties in solving NP-hard problems by means of analog computers

The first difficulty arises when one wants to encode an integer n: it is reasonable to encode it by a value of some physical variable of a magnitude close to n, but then its "physical size" n is exponential in bit-size $[\log n] + 1$. That is why in analog devices integers are often encoded by their bits, while each bit is encoded in an analog way.

Another difficulty is that many analog computers bring an underlying dynamical system to a stable state, say a local minimum of a certain target function. Examples of this are the minimal-tension surface of a soap film, or the steepest gradient of a falling stone from the hill, or a minimal energy state in the Ising spin-glass model etc. But to solve an NP-hard problem one has typically to find a global minimum, and there can be an exponential number of local minima.

Some difficulties in solving NP-hard problems by means of analog computers

The first difficulty arises when one wants to encode an integer n: it is reasonable to encode it by a value of some physical variable of a magnitude close to n, but then its "physical size" n is exponential in bit-size $[\log n] + 1$. That is why in analog devices integers are often encoded by their bits, while each bit is encoded in an analog way.

Another difficulty is that many analog computers bring an underlying dynamical system to a stable state, say a local minimum of a certain target function. Examples of this are the minimal-tension surface of a soap film, or the steepest gradient of a falling stone from the hill, or a minimal energy state in the Ising spin-glass model etc. But to solve an NP-hard problem one has typically to find a global minimum, and there can be an exponential number of local minima.

Some difficulties in solving NP-hard problems by means of analog computers

The first difficulty arises when one wants to encode an integer n: it is reasonable to encode it by a value of some physical variable of a magnitude close to n, but then its ”physical size” n is exponential in bit-size $[\log n] + 1$. That is why in analog devices integers are often encoded by their bits, while each bit is encoded in an analog way.

Another difficulty is that many analog computers bring an underlying dynamical system to a stable state, say a local minimum of a certain target function. Examples of this are the minimal-tension surface of a soap film, or the steepest gradient of a falling stone from the hill, or a minimal energy state in the Ising spin-glass model etc. But to solve an NP-hard problem one has typically to find a global minimum, and there can be an exponential number of local minima.

Some difficulties in solving NP-hard problems by means of analog computers

The first difficulty arises when one wants to encode an integer n: it is reasonable to encode it by a value of some physical variable of a magnitude close to n, but then its ”physical size” n is exponential in bit-size $[\log n] + 1$. That is why in analog devices integers are often encoded by their bits, while each bit is encoded in an analog way.

Another difficulty is that many analog computers bring an underlying dynamical system to a stable state, say a local minimum of a certain target function. Examples of this are the minimal-tension surface of a soap film, or the steepest gradient of a falling stone from the hill, or a minimal energy state in the Ising spin-glass model etc. But to solve an NP-hard problem one has typically to find a global minimum, and there can be an exponential number of local minima.

More difficulties in solving NP-hard problems by means of analog computers

For analog computers based, say on linkages one has to solve the following problem: let a device have k links, then one has to test whether some m links (for a certain m) intersect at one point? It leads to a search of $\binom{k}{m}$ combinations, and besides that to verify that these m links indeed have a common point which depends on a precision of measurements.

In other words, it is unclear how to encode combinatorics involving discrete sets of exponential size hidden in NP-hard problems with a help of a continuous analog device in an efficient manner.

There is even an opinion that impossibility of solving NP-hard problems within polynomial time by means of a physical device is a law of physics, like conservation of energy or the principle of uncertainty.

More difficulties in solving NP-hard problems by means of analog computers

For analog computers based, say on linkages one has to solve the following problem: let a device have k links, then one has to test whether some m links (for a certain m) intersect at one point? It leads to a search of $\binom{k}{m}$ combinations, and besides that to verify that these m links indeed have a common point which depends on a precision of measurements.

In other words, it is unclear how to encode combinatorics involving discrete sets of exponential size hidden in NP-hard problems with a help of a continuous analog device in an efficient manner.

There is even an opinion that impossibility of solving NP-hard problems within polynomial time by means of a physical device is a law of physics, like conservation of energy or the principle of uncertainty.

More difficulties in solving NP-hard problems by means of analog computers

For analog computers based, say on linkages one has to solve the following problem: let a device have k links, then one has to test whether some m links (for a certain m) intersect at one point? It leads to a search of $\binom{k}{m}$ combinations, and besides that to verify that these m links indeed have a common point which depends on a precision of measurements.

In other words, it is unclear how to encode combinatorics involving discrete sets of exponential size hidden in NP-hard problems with a help of a continuous analog device in an efficient manner.

There is even an opinion that impossibility of solving NP-hard problems within polynomial time by means of a physical device is a law of physics, like conservation of energy or the principle of uncertainty.

More difficulties in solving NP-hard problems by means of analog computers

For analog computers based, say on linkages one has to solve the following problem: let a device have k links, then one has to test whether some m links (for a certain m) intersect at one point? It leads to a search of $\binom{k}{m}$ combinations, and besides that to verify that these m links indeed have a common point which depends on a precision of measurements.

In other words, it is unclear how to encode combinatorics involving discrete sets of exponential size hidden in NP-hard problems with a help of a continuous analog device in an efficient manner.

There is even an opinion that impossibility of solving NP-hard problems within polynomial time by means of a physical device is a law of physics, like conservation of energy or the principle of uncertainty.

More difficulties in solving NP-hard problems by means of analog computers

For analog computers based, say on linkages one has to solve the following problem: let a device have k links, then one has to test whether some m links (for a certain m) intersect at one point? It leads to a search of $\binom{k}{m}$ combinations, and besides that to verify that these m links indeed have a common point which depends on a precision of measurements.

In other words, it is unclear how to encode combinatorics involving discrete sets of exponential size hidden in NP-hard problems with a help of a continuous analog device in an efficient manner.

There is even an opinion that impossibility of solving NP-hard problems within polynomial time by means of a physical device is a law of physics, like conservation of energy or the principle of uncertainty.

More difficulties in solving NP-hard problems by means of analog computers

For analog computers based, say on linkages one has to solve the following problem: let a device have k links, then one has to test whether some m links (for a certain m) intersect at one point? It leads to a search of $\binom{k}{m}$ combinations, and besides that to verify that these m links indeed have a common point which depends on a precision of measurements.

In other words, it is unclear how to encode combinatorics involving discrete sets of exponential size hidden in NP-hard problems with a help of a continuous analog device in an efficient manner.

There is even an opinion that impossibility of solving NP-hard problems within polynomial time by means of a physical device is a law of physics, like conservation of energy or the principle of uncertainty.

More difficulties in solving NP-hard problems by means of analog computers

For analog computers based, say on linkages one has to solve the following problem: let a device have k links, then one has to test whether some m links (for a certain m) intersect at one point? It leads to a search of $\binom{k}{m}$ combinations, and besides that to verify that these m links indeed have a common point which depends on a precision of measurements.

In other words, it is unclear how to encode combinatorics involving discrete sets of exponential size hidden in NP-hard problems with a help of a continuous analog device in an efficient manner.

There is even an opinion that impossibility of solving NP-hard problems within polynomial time by means of a physical device is a law of physics, like conservation of energy or the principle of uncertainty.

More difficulties in solving NP-hard problems by means of analog computers

For analog computers based, say on linkages one has to solve the following problem: let a device have k links, then one has to test whether some m links (for a certain m) intersect at one point? It leads to a search of $\binom{k}{m}$ combinations, and besides that to verify that these m links indeed have a common point which depends on a precision of measurements.

In other words, it is unclear how to encode combinatorics involving discrete sets of exponential size hidden in NP-hard problems with a help of a continuous analog device in an efficient manner.

There is even an opinion that impossibility of solving NP-hard problems within polynomial time by means of a physical device is a law of physics, like conservation of energy or the principle of uncertainty.

Finding a shortest path by an analog computer?

Consider an NP-hard problem in which 2 points are given in 3-dimensional space and in addition several obstacles being convex polyhedra. The problem is to find the shortest path connecting these 2 points and avoiding the obstacles (or at least the length of this path).

One can try to place a source of the light at one point and measure the time when the light reaches the second point with the hope that the light follows the shortest path according to Fermat principle.

A difficulty with this approach is that the photons propagate in all possible directions, and the intensity of a signal which reaches the second point can be too little to detect it: the intensity is proportional to the deal of trajectories with lengths close to the shortest one.

To avoid this difficulty related to an exponentially small deal of ”good” objects (photons, short trajectories and at the end the solutions of an NP-hard problem) M. Ohya, I. Volovich suggest to use quantum chaos as exponential amplifier, and show that their model can solve NP-hard problems. The question is, how is it realistic?

Finding a shortest path by an analog computer?

Consider an NP-hard problem in which 2 points are given in 3-dimensional space and in addition several obstacles being convex polyhedra. The problem is to find the shortest path connecting these 2 points and avoiding the obstacles (or at least the length of this path).

One can try to place a source of the light at one point and measure the time when the light reaches the second point with the hope that the light follows the shortest path according to Fermat principle.

A difficulty with this approach is that the photons propagate in all possible directions, and the intensity of a signal which reaches the second point can be too little to detect it: the intensity is proportional to the deal of trajectories with lengths close to the shortest one.

To avoid this difficulty related to an exponentially small deal of "good" objects (photons, short trajectories and at the end the solutions of an NP-hard problem) M. Ohya, I. Volovich suggest to use quantum chaos as exponential amplifier, and show that their model can solve NP-hard problems. The question is, how is it realistic?

Finding a shortest path by an analog computer?

Consider an NP-hard problem in which 2 points are given in 3-dimensional space and in addition several obstacles being convex polyhedra. The problem is to find the shortest path connecting these 2 points and avoiding the obstacles (or at least the length of this path).

One can try to place a source of the light at one point and measure the time when the light reaches the second point with the hope that the light follows the shortest path according to Fermat principle.

A difficulty with this approach is that the photons propagate in all possible directions, and the intensity of a signal which reaches the second point can be too little to detect it: the intensity is proportional to the deal of trajectories with lengths close to the shortest one.

To avoid this difficulty related to an exponentially small deal of "good" objects (photons, short trajectories and at the end the solutions of an NP-hard problem) M. Ohya, I. Volovich suggest to use quantum chaos as exponential amplifier, and show that their model can solve NP-hard problems. The question is, how is it realistic?

Finding a shortest path by an analog computer?

Consider an NP-hard problem in which 2 points are given in 3-dimensional space and in addition several obstacles being convex polyhedra. The problem is to find the shortest path connecting these 2 points and avoiding the obstacles (or at least the length of this path).

One can try to place a source of the light at one point and measure the time when the light reaches the second point with the hope that the light follows the shortest path according to Fermat principle.

A difficulty with this approach is that the photons propagate in all possible directions, and the intensity of a signal which reaches the second point can be too little to detect it: the intensity is proportional to the deal of trajectories with lengths close to the shortest one.

To avoid this difficulty related to an exponentially small deal of "good" objects (photons, short trajectories and at the end the solutions of an NP-hard problem) M. Ohya, I. Volovich suggest to use quantum chaos as exponential amplifier, and show that their model can solve NP-hard problems. The question is, how is it realistic?

Finding a shortest path by an analog computer?

Consider an NP-hard problem in which 2 points are given in 3-dimensional space and in addition several obstacles being convex polyhedra. The problem is to find the shortest path connecting these 2 points and avoiding the obstacles (or at least the length of this path).

One can try to place a source of the light at one point and measure the time when the light reaches the second point with the hope that the light follows the shortest path according to Fermat principle.

A difficulty with this approach is that the photons propagate in all possible directions, and the intensity of a signal which reaches the second point can be too little to detect it: the intensity is proportional to the deal of trajectories with lengths close to the shortest one.

To avoid this difficulty related to an exponentially small deal of "good" objects (photons, short trajectories and at the end the solutions of an NP-hard problem) M. Ohya, I. Volovich suggest to use quantum chaos as exponential amplifier, and show that their model can solve NP-hard problems. The question is, how is it realistic?

Finding a shortest path by an analog computer?

Consider an NP-hard problem in which 2 points are given in 3-dimensional space and in addition several obstacles being convex polyhedra. The problem is to find the shortest path connecting these 2 points and avoiding the obstacles (or at least the length of this path).

One can try to place a source of the light at one point and measure the time when the light reaches the second point with the hope that the light follows the shortest path according to Fermat principle.

A difficulty with this approach is that the photons propagate in all possible directions, and the intensity of a signal which reaches the second point can be too little to detect it: the intensity is proportional to the deal of trajectories with lengths close to the shortest one.

To avoid this difficulty related to an exponentially small deal of ”good” objects (photons, short trajectories and at the end the solutions of an NP-hard problem) M. Ohya, I. Volovich suggest to use quantum chaos as exponential amplifier, and show that their model can solve NP-hard problems. The question is, how is it realistic?

Finding a shortest path by an analog computer?

Consider an NP-hard problem in which 2 points are given in 3-dimensional space and in addition several obstacles being convex polyhedra. The problem is to find the shortest path connecting these 2 points and avoiding the obstacles (or at least the length of this path).

One can try to place a source of the light at one point and measure the time when the light reaches the second point with the hope that the light follows the shortest path according to Fermat principle.

A difficulty with this approach is that the photons propagate in all possible directions, and the intensity of a signal which reaches the second point can be too little to detect it: the intensity is proportional to the deal of trajectories with lengths close to the shortest one.

To avoid this difficulty related to an exponentially small deal of ”good” objects (photons, short trajectories and at the end the solutions of an NP-hard problem) M. Ohya, I. Volovich suggest to use quantum chaos as exponential amplifier, and show that their model can solve NP-hard problems. The question is, how is it realistic?

Finding a shortest path by an analog computer?

Consider an NP-hard problem in which 2 points are given in 3-dimensional space and in addition several obstacles being convex polyhedra. The problem is to find the shortest path connecting these 2 points and avoiding the obstacles (or at least the length of this path).

One can try to place a source of the light at one point and measure the time when the light reaches the second point with the hope that the light follows the shortest path according to Fermat principle.

A difficulty with this approach is that the photons propagate in all possible directions, and the intensity of a signal which reaches the second point can be too little to detect it: the intensity is proportional to the deal of trajectories with lengths close to the shortest one.

To avoid this difficulty related to an exponentially small deal of ”good” objects (photons, short trajectories and at the end the solutions of an NP-hard problem) M. Ohya, I. Volovich suggest to use quantum chaos as exponential amplifier, and show that their model can solve NP-hard problems. The question is, how is it realistic?

Finding a shortest path by an analog computer?

Consider an NP-hard problem in which 2 points are given in 3-dimensional space and in addition several obstacles being convex polyhedra. The problem is to find the shortest path connecting these 2 points and avoiding the obstacles (or at least the length of this path).

One can try to place a source of the light at one point and measure the time when the light reaches the second point with the hope that the light follows the shortest path according to Fermat principle.

A difficulty with this approach is that the photons propagate in all possible directions, and the intensity of a signal which reaches the second point can be too little to detect it: the intensity is proportional to the deal of trajectories with lengths close to the shortest one.

To avoid this difficulty related to an exponentially small deal of ”good” objects (photons, short trajectories and at the end the solutions of an NP-hard problem) M. Ohya, I. Volovich suggest to use quantum chaos as exponential amplifier, and show that their model can solve NP-hard problems. The question is, how is it realistic?

Finding a shortest path by an analog computer?

Consider an NP-hard problem in which 2 points are given in 3-dimensional space and in addition several obstacles being convex polyhedra. The problem is to find the shortest path connecting these 2 points and avoiding the obstacles (or at least the length of this path).

One can try to place a source of the light at one point and measure the time when the light reaches the second point with the hope that the light follows the shortest path according to Fermat principle.

A difficulty with this approach is that the photons propagate in all possible directions, and the intensity of a signal which reaches the second point can be too little to detect it: the intensity is proportional to the deal of trajectories with lengths close to the shortest one.

To avoid this difficulty related to an exponentially small deal of ”good” objects (photons, short trajectories and at the end the solutions of an NP-hard problem) M. Ohya, I. Volovich suggest to use quantum chaos as exponential amplifier, and show that their model can solve NP-hard problems. The question is, how is it realistic?

Quantum computers

Some hope in solving hard combinatorial problems has emerged with appearance of the theory of quantum computers (leaving aside the difficulties predicted by R. Feynman of their practical realizing, which is still far from being executed). Better to speak more precisely about the model of D. Deutsch (it has some predecessors like an idea conceived by Yu. Manin).

The model of D.Deutsch consists in application of a unitary operator to a normalized vector. The basis of the space of vectors consists in n-tuples of the states (say, 0 or 1) of what is called q(uantum)-bits. Thus, the dimension of the space is 2^n. For any vector v a complex coefficients a of its expansion (as a linear combination of basis vectors) at a particular basis vector e is called the *amplitude* of e in v, and $|c|^2$ equals the probability of appearance e while (quantum) observation of v.

We mention that also adiabatic machines were studied, and A. Kitaev has proved their equivalence (from the complexity point of view) to the discussed model of D. Deutsch.

Quantum computers

Some hope in solving hard combinatorial problems has emerged with appearance of the theory of quantum computers (leaving aside the difficulties predicted by R. Feynman of their practical realizing, which is still far from being executed). Better to speak more precisely about the model of D. Deutsch (it has some predecessors like an idea conceived by Yu. Manin).

The model of D.Deutsch consists in application of a unitary operator to a normalized vector. The basis of the space of vectors consists in n-tuples of the states (say, 0 or 1) of what is called q(uantum)-bits. Thus, the dimension of the space is 2^n. For any vector v a complex coefficients a of its expansion (as a linear combination of basis vectors) at a particular basis vector e is called the *amplitude* of e in v, and $|c|^2$ equals the probability of appearance e while (quantum) observation of v.

We mention that also adiabatic machines were studied, and A. Kitaev has proved their equivalence (from the complexity point of view) to the discussed model of D. Deutsch.

Quantum computers

Some hope in solving hard combinatorial problems has emerged with appearance of the theory of quantum computers (leaving aside the difficulties predicted by R. Feynman of their practical realizing, which is still far from being executed). Better to speak more precisely about the model of D. Deutsch (it has some predecessors like an idea conceived by Yu. Manin).

The model of D.Deutsch consists in application of a unitary operator to a normalized vector. The basis of the space of vectors consists in n-tuples of the states (say, 0 or 1) of what is called q(uantum)-bits. Thus, the dimension of the space is 2^n. For any vector v a complex coefficients a of its expansion (as a linear combination of basis vectors) at a particular basis vector e is called the *amplitude* of e in v, and $|c|^2$ equals the probability of appearance e while (quantum) observation of v.

We mention that also adiabatic machines were studied, and A. Kitaev has proved their equivalence (from the complexity point of view) to the discussed model of D. Deutsch.

Quantum computers

Some hope in solving hard combinatorial problems has emerged with appearance of the theory of quantum computers (leaving aside the difficulties predicted by R. Feynman of their practical realizing, which is still far from being executed). Better to speak more precisely about the model of D. Deutsch (it has some predecessors like an idea conceived by Yu. Manin).

The model of D.Deutsch consists in application of a unitary operator to a normalized vector. The basis of the space of vectors consists in n-tuples of the states (say, 0 or 1) of what is called q(uantum)-bits. Thus, the dimension of the space is 2^n. For any vector v a complex coefficients a of its expansion (as a linear combination of basis vectors) at a particular basis vector e is called the *amplitude* of e in v, and $|c|^2$ equals the probability of appearance e while (quantum) observation of v.

We mention that also adiabatic machines were studied, and A. Kitaev has proved their equivalence (from the complexity point of view) to the discussed model of D. Deutsch.

Quantum computers

Some hope in solving hard combinatorial problems has emerged with appearance of the theory of quantum computers (leaving aside the difficulties predicted by R. Feynman of their practical realizing, which is still far from being executed). Better to speak more precisely about the model of D. Deutsch (it has some predecessors like an idea conceived by Yu. Manin).

The model of D.Deutsch consists in application of a unitary operator to a normalized vector. The basis of the space of vectors consists in n-tuples of the states (say, 0 or 1) of what is called q(uantum)-bits. Thus, the dimension of the space is 2^n. For any vector v a complex coefficients a of its expansion (as a linear combination of basis vectors) at a particular basis vector e is called the *amplitude* of e in v, and $|c|^2$ equals the probability of appearance e while (quantum) observation of v.

We mention that also adiabatic machines were studied, and A. Kitaev has proved their equivalence (from the complexity point of view) to the discussed model of D. Deutsch.

Quantum computers

Some hope in solving hard combinatorial problems has emerged with appearance of the theory of quantum computers (leaving aside the difficulties predicted by R. Feynman of their practical realizing, which is still far from being executed). Better to speak more precisely about the model of D. Deutsch (it has some predecessors like an idea conceived by Yu. Manin).

The model of D.Deutsch consists in application of a unitary operator to a normalized vector. The basis of the space of vectors consists in n-tuples of the states (say, 0 or 1) of what is called q(uantum)-bits. Thus, the dimension of the space is 2^n. For any vector v a complex coefficients a of its expansion (as a linear combination of basis vectors) at a particular basis vector e is called the *amplitude* of e in v, and $|c|^2$ equals the probability of appearance e while (quantum) observation of v.

We mention that also adiabatic machines were studied, and A. Kitaev has proved their equivalence (from the complexity point of view) to the discussed model of D. Deutsch.

Quantum computers

Some hope in solving hard combinatorial problems has emerged with appearance of the theory of quantum computers (leaving aside the difficulties predicted by R. Feynman of their practical realizing, which is still far from being executed). Better to speak more precisely about the model of D. Deutsch (it has some predecessors like an idea conceived by Yu. Manin).

The model of D.Deutsch consists in application of a unitary operator to a normalized vector. The basis of the space of vectors consists in n-tuples of the states (say, 0 or 1) of what is called q(uantum)-bits. Thus, the dimension of the space is 2^n. For any vector v a complex coefficients a of its expansion (as a linear combination of basis vectors) at a particular basis vector e is called the *amplitude* of e in v, and $|c|^2$ equals the probability of appearance e while (quantum) observation of v.

We mention that also adiabatic machines were studied, and A. Kitaev has proved their equivalence (from the complexity point of view) to the discussed model of D. Deutsch.

Quantum computers

Some hope in solving hard combinatorial problems has emerged with appearance of the theory of quantum computers (leaving aside the difficulties predicted by R. Feynman of their practical realizing, which is still far from being executed). Better to speak more precisely about the model of D. Deutsch (it has some predecessors like an idea conceived by Yu. Manin).

The model of D.Deutsch consists in application of a unitary operator to a normalized vector. The basis of the space of vectors consists in n-tuples of the states (say, 0 or 1) of what is called q(uantum)-bits. Thus, the dimension of the space is 2^n. For any vector v a complex coefficients a of its expansion (as a linear combination of basis vectors) at a particular basis vector e is called the *amplitude* of e in v, and $|c|^2$ equals the probability of appearance e while (quantum) observation of v.

We mention that also adiabatic machines were studied, and A. Kitaev has proved their equivalence (from the complexity point of view) to the discussed model of D. Deutsch.

Quantum computers

Some hope in solving hard combinatorial problems has emerged with appearance of the theory of quantum computers (leaving aside the difficulties predicted by R. Feynman of their practical realizing, which is still far from being executed). Better to speak more precisely about the model of D. Deutsch (it has some predecessors like an idea conceived by Yu. Manin).

The model of D.Deutsch consists in application of a unitary operator to a normalized vector. The basis of the space of vectors consists in n-tuples of the states (say, 0 or 1) of what is called q(uantum)-bits. Thus, the dimension of the space is 2^n. For any vector v a complex coefficients a of its expansion (as a linear combination of basis vectors) at a particular basis vector e is called the *amplitude* of e in v, and $|c|^2$ equals the probability of appearance e while (quantum) observation of v.

We mention that also adiabatic machines were studied, and A. Kitaev has proved their equivalence (from the complexity point of view) to the discussed model of D. Deutsch.

Quantum computers

Some hope in solving hard combinatorial problems has emerged with appearance of the theory of quantum computers (leaving aside the difficulties predicted by R. Feynman of their practical realizing, which is still far from being executed). Better to speak more precisely about the model of D. Deutsch (it has some predecessors like an idea conceived by Yu. Manin).
The model of D.Deutsch consists in application of a unitary operator to a normalized vector. The basis of the space of vectors consists in n-tuples of the states (say, 0 or 1) of what is called q(uantum)-bits. Thus, the dimension of the space is 2^n. For any vector v a complex coefficients a of its expansion (as a linear combination of basis vectors) at a particular basis vector e is called the *amplitude* of e in v, and $|c|^2$ equals the probability of appearance e while (quantum) observation of v.

We mention that also adiabatic machines were studied, and A. Kitaev has proved their equivalence (from the complexity point of view) to the discussed model of D. Deutsch.

Shor's factoring algorithm and its menacing consequences for cryptography

The famous quantum algorithm of P. Shor can factor integers within polynomial time. Since the security of the overwhelming part of the modern practical cryptography, like in bank cards, relies on the presumable difficulty of factoring, an eventual design of quantum computers would break the modern cryptography and force to change cryptosystems considerably, but still the design of quantum computers is far from realization.

It is an open question whether quantum computers can solve NP-hard problems within polynomial time? The best result in this direction is Grover's algorithm which allows one to speed-up solving NP-hard problems by the square root.

In fact, quantum computers have features of both analog computers (since they involve observations) and of digital computers (since they deal with q-bits and with their discrete states after observations). Quantum computers are equivalent to Turing machines up to exponential time.

Shor's factoring algorithm and its menacing consequences for cryptography

The famous quantum algorithm of P. Shor can factor integers within polynomial time. Since the security of the overwhelming part of the modern practical cryptography, like in bank cards, relies on the presumable difficulty of factoring, an eventual design of quantum computers would break the modern cryptography and force to change cryptosystems considerably, but still the design of quantum computers is far from realization.

It is an open question whether quantum computers can solve NP-hard problems within polynomial time? The best result in this direction is Grover's algorithm which allows one to speed-up solving NP-hard problems by the square root.

In fact, quantum computers have features of both analog computers (since they involve observations) and of digital computers (since they deal with q-bits and with their discrete states after observations). Quantum computers are equivalent to Turing machines up to exponential time.

Shor's factoring algorithm and its menacing consequences for cryptography

The famous quantum algorithm of P. Shor can factor integers within polynomial time. Since the security of the overwhelming part of the modern practical cryptography, like in bank cards, relies on the presumable difficulty of factoring, an eventual design of quantum computers would break the modern cryptography and force to change cryptosystems considerably, but still the design of quantum computers is far from realization.

It is an open question whether quantum computers can solve NP-hard problems within polynomial time? The best result in this direction is Grover's algorithm which allows one to speed-up solving NP-hard problems by the square root.

In fact, quantum computers have features of both analog computers (since they involve observations) and of digital computers (since they deal with q-bits and with their discrete states after observations). Quantum computers are equivalent to Turing machines up to exponential time.

Shor's factoring algorithm and its menacing consequences for cryptography

The famous quantum algorithm of P. Shor can factor integers within polynomial time. Since the security of the overwhelming part of the modern practical cryptography, like in bank cards, relies on the presumable difficulty of factoring, an eventual design of quantum computers would break the modern cryptography and force to change cryptosystems considerably, but still the design of quantum computers is far from realization.

It is an open question whether quantum computers can solve NP-hard problems within polynomial time? The best result in this direction is Grover's algorithm which allows one to speed-up solving NP-hard problems by the square root.

In fact, quantum computers have features of both analog computers (since they involve observations) and of digital computers (since they deal with *q*-bits and with their discrete states after observations). Quantum computers are equivalent to Turing machines up to exponential time.

Shor's factoring algorithm and its menacing consequences for cryptography

The famous quantum algorithm of P. Shor can factor integers within polynomial time. Since the security of the overwhelming part of the modern practical cryptography, like in bank cards, relies on the presumable difficulty of factoring, an eventual design of quantum computers would break the modern cryptography and force to change cryptosystems considerably, but still the design of quantum computers is far from realization.

It is an open question whether quantum computers can solve NP-hard problems within polynomial time? The best result in this direction is Grover's algorithm which allows one to speed-up solving NP-hard problems by the square root.

In fact, quantum computers have features of both analog computers (since they involve observations) and of digital computers (since they deal with q-bits and with their discrete states after observations). Quantum computers are equivalent to Turing machines up to exponential time.

Shor's factoring algorithm and its menacing consequences for cryptography

The famous quantum algorithm of P. Shor can factor integers within polynomial time. Since the security of the overwhelming part of the modern practical cryptography, like in bank cards, relies on the presumable difficulty of factoring, an eventual design of quantum computers would break the modern cryptography and force to change cryptosystems considerably, but still the design of quantum computers is far from realization.

It is an open question whether quantum computers can solve NP-hard problems within polynomial time? The best result in this direction is Grover's algorithm which allows one to speed-up solving NP-hard problems by the square root.

In fact, quantum computers have features of both analog computers (since they involve observations) and of digital computers (since they deal with *q*-bits and with their discrete states after observations). Quantum computers are equivalent to Turing machines up to exponential time.

Shor's factoring algorithm and its menacing consequences for cryptography

The famous quantum algorithm of P. Shor can factor integers within polynomial time. Since the security of the overwhelming part of the modern practical cryptography, like in bank cards, relies on the presumable difficulty of factoring, an eventual design of quantum computers would break the modern cryptography and force to change cryptosystems considerably, but still the design of quantum computers is far from realization.

It is an open question whether quantum computers can solve NP-hard problems within polynomial time? The best result in this direction is Grover's algorithm which allows one to speed-up solving NP-hard problems by the square root.

In fact, quantum computers have features of both analog computers (since they involve observations) and of digital computers (since they deal with q-bits and with their discrete states after observations). Quantum computers are equivalent to Turing machines up to exponential time.

Shor's factoring algorithm and its menacing consequences for cryptography

The famous quantum algorithm of P. Shor can factor integers within polynomial time. Since the security of the overwhelming part of the modern practical cryptography, like in bank cards, relies on the presumable difficulty of factoring, an eventual design of quantum computers would break the modern cryptography and force to change cryptosystems considerably, but still the design of quantum computers is far from realization.

It is an open question whether quantum computers can solve NP-hard problems within polynomial time? The best result in this direction is Grover's algorithm which allows one to speed-up solving NP-hard problems by the square root.

In fact, quantum computers have features of both analog computers (since they involve observations) and of digital computers (since they deal with q-bits and with their discrete states after observations).

Quantum computers are equivalent to Turing machines up to exponential time.

Shor's factoring algorithm and its menacing consequences for cryptography

The famous quantum algorithm of P. Shor can factor integers within polynomial time. Since the security of the overwhelming part of the modern practical cryptography, like in bank cards, relies on the presumable difficulty of factoring, an eventual design of quantum computers would break the modern cryptography and force to change cryptosystems considerably, but still the design of quantum computers is far from realization.

It is an open question whether quantum computers can solve NP-hard problems within polynomial time? The best result in this direction is Grover's algorithm which allows one to speed-up solving NP-hard problems by the square root.

In fact, quantum computers have features of both analog computers (since they involve observations) and of digital computers (since they deal with *q*-bits and with their discrete states after observations). Quantum computers are equivalent to Turing machines up to exponential time.

Quantum cryptography

Quantum computers suggest an alternative approach to cryptography: the well-known cryptosystem by C. Benneth, G. Brassard. It proposes a completely different approach to transmitting secrets via a public channel.

In a classical cryptography the paradigm is to transmit an encoded message in such a way that an adversary observing the transmission in a public channel would not be able to reveal the original message. While quantum cryptography makes use of the quantum phenomenon that just the observation of a q-bit destroys its state, and therefore the receiver of the message can detect that an adversary has made an observation and abort the communicating session. Then they start a new session.

Thus, the quantum cryptography is not quite reliable because an adversary can observe all the sessions and completely prevent from communicating.

Quantum cryptography

Quantum computers suggest an alternative approach to cryptography: the well-known cryptosystem by C. Benneth, G. Brassard. It proposes a completely different approach to transmitting secrets via a public channel.

In a classical cryptography the paradigm is to transmit an encoded message in such a way that an adversary observing the transmission in a public channel would not be able to reveal the original message. While quantum cryptography makes use of the quantum phenomenon that just the observation of a *q*-bit destroys its state, and therefore the receiver of the message can detect that an adversary has made an observation and abort the communicating session. Then they start a new session.

Thus, the quantum cryptography is not quite reliable because an adversary can observe all the sessions and completely prevent from communicating.

Quantum cryptography

Quantum computers suggest an alternative approach to cryptography: the well-known cryptosystem by C. Benneth, G. Brassard. It proposes a completely different approach to transmitting secrets via a public channel.

In a classical cryptography the paradigm is to transmit an encoded message in such a way that an adversary observing the transmission in a public channel would not be able to reveal the original message. While quantum cryptography makes use of the quantum phenomenon that just the observation of a *q*-bit destroys its state, and therefore the receiver of the message can detect that an adversary has made an observation and abort the communicating session. Then they start a new session.

Thus, the quantum cryptography is not quite reliable because an adversary can observe all the sessions and completely prevent from communicating.

Quantum cryptography

Quantum computers suggest an alternative approach to cryptography: the well-known cryptosystem by C. Benneth, G. Brassard. It proposes a completely different approach to transmitting secrets via a public channel.

In a classical cryptography the paradigm is to transmit an encoded message in such a way that an adversary observing the transmission in a public channel would not be able to reveal the original message. While quantum cryptography makes use of the quantum phenomenon that just the observation of a q-bit destroys its state, and therefore the receiver of the message can detect that an adversary has made an observation and abort the communicating session. Then they start a new session.

Thus, the quantum cryptography is not quite reliable because an adversary can observe all the sessions and completely prevent from communicating.

Quantum cryptography

Quantum computers suggest an alternative approach to cryptography: the well-known cryptosystem by C. Benneth, G. Brassard. It proposes a completely different approach to transmitting secrets via a public channel.

In a classical cryptography the paradigm is to transmit an encoded message in such a way that an adversary observing the transmission in a public channel would not be able to reveal the original message. While quantum cryptography makes use of the quantum phenomenon that just the observation of a *q*-bit destroys its state, and therefore the receiver of the message can detect that an adversary has made an observation and abort the communicating session. Then they start a new session.

Thus, the quantum cryptography is not quite reliable because an adversary can observe all the sessions and completely prevent from communicating.

Quantum cryptography

Quantum computers suggest an alternative approach to cryptography: the well-known cryptosystem by C. Benneth, G. Brassard. It proposes a completely different approach to transmitting secrets via a public channel.

In a classical cryptography the paradigm is to transmit an encoded message in such a way that an adversary observing the transmission in a public channel would not be able to reveal the original message. While quantum cryptography makes use of the quantum phenomenon that just the observation of a *q*-bit destroys its state, and therefore the receiver of the message can detect that an adversary has made an observation and abort the communicating session. Then they start a new session.

Thus, the quantum cryptography is not quite reliable because an adversary can observe all the sessions and completely prevent from communicating.

Quantum cryptography

Quantum computers suggest an alternative approach to cryptography: the well-known cryptosystem by C. Benneth, G. Brassard. It proposes a completely different approach to transmitting secrets via a public channel.

In a classical cryptography the paradigm is to transmit an encoded message in such a way that an adversary observing the transmission in a public channel would not be able to reveal the original message. While quantum cryptography makes use of the quantum phenomenon that just the observation of a q-bit destroys its state, and therefore the receiver of the message can detect that an adversary has made an observation and abort the communicating session. Then they start a new session.

Thus, the quantum cryptography is not quite reliable because an adversary can observe all the sessions and completely prevent from communicating.

Quantum teleportation

Another application of quantum devices which was successfully realized, is a teleportation. It is based on the quantum phenomenon (it is called Einstein-Podolsky-Rosen paradox) that if there is a pair of entangled photons and one of them is sent to a remote receiver then the observation of the state of one photon determines the state of its remote counterpart.

Perhaps, this can be used for remote communications and for cryptography.

Quantum teleportation

Another application of quantum devices which was successfully realized, is a teleportation. It is based on the quantum phenomenon (it is called Einstein-Podolsky-Rosen paradox) that if there is a pair of entangled photons and one of them is sent to a remote receiver then the observation of the state of one photon determines the state of its remote counterpart.

Perhaps, this can be used for remote communications and for cryptography.

Quantum teleportation

Another application of quantum devices which was successfully realized, is a teleportation. It is based on the quantum phenomenon (it is called Einstein-Podolsky-Rosen paradox) that if there is a pair of entangled photons and one of them is sent to a remote receiver then the observation of the state of one photon determines the state of its remote counterpart.

Perhaps, this can be used for remote communications and for cryptography.

Quantum teleportation

Another application of quantum devices which was successfully realized, is a teleportation. It is based on the quantum phenomenon (it is called Einstein-Podolsky-Rosen paradox) that if there is a pair of entangled photons and one of them is sent to a remote receiver then the observation of the state of one photon determines the state of its remote counterpart.

Perhaps, this can be used for remote communications and for cryptography.

Quantum teleportation

Another application of quantum devices which was successfully realized, is a teleportation. It is based on the quantum phenomenon (it is called Einstein-Podolsky-Rosen paradox) that if there is a pair of entangled photons and one of them is sent to a remote receiver then the observation of the state of one photon determines the state of its remote counterpart.

Perhaps, this can be used for remote communications and for cryptography.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography and assumption of existence of one-way functions

While the future of analog computations is not quite clear, their applications in cryptography look more prospective.

In the conventional cryptography when Bob wants to transmit his secret message m to Alice via a public channel, he fulfils it by means of a one-way function f for encoding m. Informally speaking, it is easy to compute $c = f(m)$, where c is the code transmitted via the public channel, but on the other hand, for an adversary who observes just c it is presumably difficult to restore m.

Moreover, just the existence of a cryptosystem implies the existence of a one-way function, thus one-way functions are inevitable in the conventional cryptography.

In its turn, the existence of a one-way function would entail that $P \neq NP$, therefore a proof of existence of a one-way function is unlikely. Thus, the security of the conventional cryptography relies on an unproved assumption.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Cryptography based on physical principles

Jointly with V. Shpilrain we have suggested cryptosystems based just on physical laws, so their security does not depend on any mathematical assumptions. I describe roughly the simplest (but not the most efficient) among them just to explain a basic idea.

Let Alice and Bob communicate via a public wave channel, and they have agreed in advance that they emit waves of the agreed frequency and phase. If Bob wants to transmit a (secret message) integer m he emits a wave with amplitude m, while Alice emits a wave with some amplitude a known only to her. As a result of the interference the resulting wave in the channel will be of amplitude $m + a$ (and of their common frequency and phase). Alice restores the secret message m knowing resulting amplitude $m + a$ and her own a, while an adversary just observing $m + a$, is unable to find m.

In other words, unlike the conventional cryptography when the receiver (Alice) is passive just reading what Bob has sent her via the public channel, in the physical-based cryptography Alice is actively influencing on the contents of the channel.

Efficiency of cryptosystems based on physical principles

The described cryptosystem seems to be more efficient than the conventional ones because it does not require calculation of a one-way function, but on the other hand, as always with analog computers,it depends on the precision of measurements.

Philosophical conclusion

Perhaps, the people will come back to analog computers, but in a higher level of development on the dialectics spiral.

Efficiency of cryptosystems based on physical principles

The described cryptosystem seems to be more efficient than the conventional ones because it does not require calculation of a one-way function, but on the other hand, as always with analog computers,it depends on the precision of measurements.

Philosophical conclusion

Perhaps, the people will come back to analog computers, but in a higher level of development on the dialectics spiral.

Efficiency of cryptosystems based on physical principles

The described cryptosystem seems to be more efficient than the conventional ones because it does not require calculation of a one-way function, but on the other hand, as always with analog computers,it depends on the precision of measurements.

Philosophical conclusion

Perhaps, the people will come back to analog computers, but in a higher level of development on the dialectics spiral.